CONTROL DATA
CORPORATION

CONTROL DATA®
CYBER 70 MODEL 74
COMPUTER SYSTEM

SYSTEM DESCRIPTION AND PROGRAMMING INFORMATION REFERENCE MANUAL VOLUME 1

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released.
(2-22-71)	
B	Manual revised. Addition of Interlock Register information.
(3-6-71)	
C	Addition of timing information and minor changes. ECO 27834.
(7-71)	
D	Manual revised; Engineering Change Order 29919, publication change only. Pages v, 2-1, 2-34,
(11-16-71)	and Comment Sheet revised.
E	Manual revised; Engineering Change Order 33547. Pages v, 1-2, 1-9, 1-10, 1-11, 1-12, 2-7, 2-8,
(4-13-73)	2-9, 2-10, 2-11, 2-12, 2-17, 2-18, 2-19, 2-31, 2-32, 2-35, and 2-36 are revised. Pages 2-8.1,
	11-.1, and 32.1 are added.
F	
(4-1-74)	Manual revised; includes Engineering Change Order 33534. Pages 2-10 and 2-35 are revised.
G	Manual revised; includes Field Change Order 34719. Pages 2-24 and 2-31 are revised.
(6-26-74)	
H	Manual revised; includes Engineering Change Order 35407, publication change only. Pages 2-34
(8-7-74)	and 2-35 are revised.
J	Manual revised; includes Engineering Change Order 35429. Pages 2-8, 2-8.1, 2-9 and 2-10 are
(9-9-74)	revised.
K	Manual revised; includes Engineering Change Order 35678. Pages 2-12, 2-13, 2-18 and 2-19 are
(12-5-74)	revised.
L	Manual revised; includes Engineering Change Order 36380. Page 1-10 is revised.
(6-24-75)	
M	Manual revised; includes Engineering Change Order 37148. Pages ii, 2-14, and 2-28 are revised.
(6-29-76)	
N	Manual revised; includes Engineering Change Order 37836. Pages iii, 2-21, and 2-22 are revised.
(6-15-77)	

Publication No. 60347400

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

Printed in the United States of America

PREFACE

The CONTROL DATA® CYBER 70 series reference manuals are published in a series of volumes. This manual is volume 1 of the series.

This volume contains the Systems Description and general programming information. Volume 2, publication number 60347300, contains detailed descriptions of the central processor and the peripheral processor instructions.

Information about the ECS (Extended Core Storage option) is in volume 3 of the series, publication number 60347100.

The publications listed are available through the nearest Control Data Corporation sales office.

CONTENTS

1. SYSTEM DESCRIPTION

Introduction 1-1
System Characteristics 1-3
- Central Processor Characteristics 1-3
- Peripheral Processor Characteristics 1-3
- Central Memory Characteristics 1-3

Functional Descriptions 1-3
- Central Processor 1-3
- Peripheral Processors 1-7
- Central Memory 1-10

2. PROGRAMMING INFORMATION

Central Processor Programming 2-1
- Instruction Formats 2-1
- Operating Registers 2-2
- Program Address Register 2-4
- Exchange Jump 2-5
- Reference Address 2-7
- Exit Mode 2-7
- Floating Point Arithmetic 2-11.1
- Fixed Point Arithmetic 2-19

Peripheral Processor Programming 2-20
- Instruction Formats 2-20
- Address Modes 2-21
- Access to Central Memory 2-22
- Input and Output 2-24
- Interlock Register 2-28

Manual Control 2-29
- Dead Start Panel 2-30
- Console 2-31

System Interrupt 2-32.1
- Hardware Provisions for Interrupt 2-32.1

Timing Information 2-33
- Central Processor Timing 2-33
- Peripheral Processor Timing 2-36

FIGURES

1-1 MODEL 74-YZ Mainframe 1-2
1-2 Memory Map 1-12
2-1 Central Processor Instruction Format 2-2
2-2 Central Processor Operating Registers 2-3
2-3 Exchange Jump Package 2-5
2-5 Console Operator Control Panel 2-32

TABLES

2-1 Exit Mode: Address Out of Bounds 2-9
2-1.1 Unconditional Exit Actions 2-10
2-2 Range of Permissible Exponents 2-12
2-3 Indefinite Forms 2-13
2-4 Overflow and Underflow Conditions 2-16
2-5 Central Processor Instruction Execution Times 2-34
2-6 Peripheral Processor Instruction Execution Times 2-37

SYSTEM DESCRIPTION 1.

INTRODUCTION

The CONTROL DATA® CYBER 70 MODEL 74-YZ Computer Systems consist of a mainframe and a flexible assortment of peripheral and control equipment. A system usually will have a control console and input/output devices such as stations, card readers, magnetic tape drives, mass storage units and printers. Extended Core Storage (ECS) offered in a variety of sizes may be used to augment the system.

The mainframe contains 10, 14, 17, or 20 peripheral processors (PPU's) and the data channels necessary to communicate with the peripheral equipment. A central memory (CM), a central processor unit (CPU), with 24 operating registers per arithmetic unit, (one or two) and the attendant control logic are the major components on the mainframe. Optional couplers or controllers may be in the mainframe on some systems. Figure 1-1 shows the mainframe and some of the optional equipment.

The contents of this manual are concerned with the basic system without attempting to describe or give programming information for the peripheral equipment. The peripheral equipment and their controllers are covered in separate manuals.

The system model numbers are assigned as follows:

MODEL 74-YZ

- System Model
- Code for number of Central Processors in the system. (either 1 or 2)
- Code for the size of the Central Memory*

*CENTRAL MEMORY CODES

CODE	SIZE (60-Bit Words)
2	32K
3	49K
4	65K
6	98K
8	131K

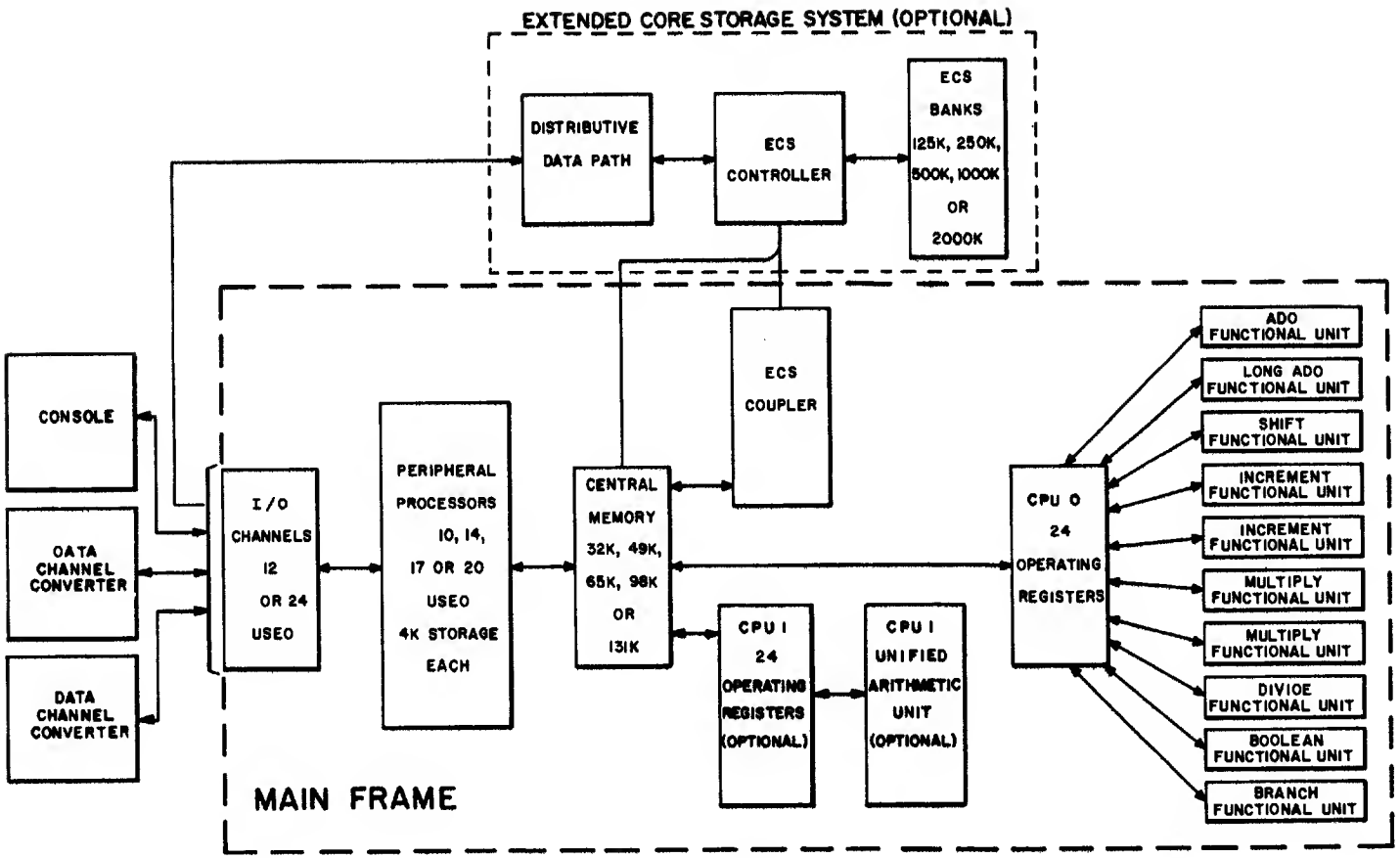


Figure 1-1. MODEL 74-YZ Mainframe

SYSTEM CHARACTERISTICS

CENTRAL PROCESSOR CHARACTERISTICS

- 60-bit word length
- Computation in floating point and fixed point, single and double precision
- 24 operating registers per central processor
- 10 arithmetic functional units for concurrent operations
- Memory transfer rate of up to one word each 100 nsec
- Instruction stack which holds up to 27 instructions for simplified reference access
- Optional dual processor configuration (second processor is a unified arithmetic unit.)

PERIPHERAL PROCESSOR CHARACTERISTICS

- 12-bit word length
- Computation in fixed point
- Time-shared access to central memory
- Internal memory of 4,096 12-bit words
- 10, 14, 17, or 20 processors

CENTRAL MEMORY CHARACTERISTICS

- Capacity of 32,768 to 131,072 60-bit words
- Independent bank construction, to allow separate access to each 4K bank of memory (called phasing)
- Transfer rate up to 1 word each 100 nsec in phased operation

FUNCTIONAL DESCRIPTIONS

CENTRAL PROCESSOR

The central processor is an arithmetic processor which communicates only with central memory. It is isolated from the peripheral processors and is thus free to carry on computation unencumbered by input/output requirements. It consists of 10 functional units and control logic. The functional units contain all logic necessary to execute the arithmetic, manipulative and logical operations. The control logic directs the arithmetic operations and provides the interface between the functional units and central memory. The control logic

also performs instruction retrieving, address preparation, memory protection, and data retrieving and storing. The second processor is a unified arithmetic unit. It performs much the same as the functional units except that it operates serially.

PROGRAM HARDWARE RELATIONSHIPS

Programs for the central processor are held in central memory. A program is started with an Exchange Jump instruction from a peripheral processor. The Exchange Jump instruction specifies the location in central memory of the central processor program, specifies the mode of exit (normal or error) for the program, and sets initial quantities in the operating registers.

OPERATING REGISTERS

Twenty-four operating registers are provided to mimimize memory references:

- 8 address registers, 18 bits in length
- 8 increment registers, 18 bits in length
- 8 operand registers, 60 bits in length

PROGRAM HANDLING

Programs are written for the central processors in a conventional manner, specifying a sequence of arithmetic and control operations. Each instruction in a program is brought up in its turn from one of the instruction registers. These registers are filled from central memory. Each central processor is programmed independently.

BRANCHING

A branch to another area of the program voids the previous instructions in the registers and brings in new instructions. When a new instruction is brought up, a test is made, to determine if the arithmetic unit is busy, or if reservation conflict is possible. If the unit is free and no conflict is present, the entire instruction is given to the arithmetic unit for further action. Another instruction may then be brought up and issued.

PROGRAM SEQUENCES

The original sequence of the program is established at the time each instruction is issued. Only those operations which depend on previous results prevent the issuing of instructions, and then only if the steps are incomplete. The reservation control keeps a running account of the address, increment, and operand registers in order to preserve the original sequence. On occasion, a program may use an Increment Store instruction to modify the contents of a memory location holding a subsequent instruction.

PROGRAM REFERENCES

Nearly all central memory references for information or instructions are made on an implicit or secondary basis. Instructions are retrieved from memory only if the instruction registers are nearly empty (or when ordered by a branch). Information is brought to or from the operand registers only when appropriate address registers are referenced during the course of a program. Such references are also accounted for in the reservation control.

PROGRAM MEMORY LOCATIONS

All central processor references to central memory are made relative to the lower boundary address. A central processor program may therefore be relocated in central memory by modifying the boundaries only. Any attempt by the central processor to reference memory outside of its boundaries causes an immediate exit which can be readily examined by a peripheral processor and displayed for the operator.

THE CENTRAL EXCHANGE JUMP

The exchange jump can be performed unconditionally (regardless of the state of the monitor flag) by the central processor. If the monitor flag is clear the jump is to the Monitor Address, or if the flag is set the address is formed by adding Bj to K. The peripheral processors also perform exchange jumps as explained in the peripheral processor section of this manual.

FUNCTIONAL UNITS

There are eight types of functional units which make up the total of ten functional units in the basic central processor. There are two multiply and two increment units. They are duplexed in operation so that when one is busy, the other can be concurrently utilized. The second central processor is a unified arithmetic type and does not have functional units. All arithmetic is performed serially in that type of processor.

ADD FUNCTIONAL UNIT

This unit performs addition and subtraction on floating point numbers or on rounded floating point numbers.

LONG ADD FUNCTIONAL UNIT

This unit performs one's complement addition and subtraction of 60-bit fixed point numbers.

SHIFT FUNCTIONAL UNIT

This unit performs left (circular) shifting, right (end-off sign extension) shifting, normalize, pack, and unpack operations. The unit also performs mask generation.

INCREMENT FUNCTIONAL UNITS

These units perform one's complement addition and subtraction of 18-bit numbers.

MULTIPLY FUNCTIONAL UNITS

The units perform multiplication on fixed point numbers, floating point numbers, or on rounded floating point numbers.

DIVIDE FUNCTIONAL UNIT

This unit performs division on floating point numbers or on rounded floating point numbers. The unit also counts the number of one's in a word.

BOOLEAN FUNCTIONAL UNIT

This unit performs the logical operations; transfer, logical product, logical sum, and logical difference.

BRANCH FUNCTIONAL UNIT

This unit performs all jumps or branches from the programs.

RESERVATION CONTROL

Special control logic is included to coordinate the interaction of the functional units with instructions and the operating registers. Conflicts can arise if several instructions simultaneously call for the use of the same functional unit or of the same operating registers. The reservation control solves these conflicts and reserves access in accordance with the solution. The reservation control logic is frequently referred to as the "scoreboard".

PERIPHERAL PROCESSORS

The peripheral processors are identical. They operate independently and simultaneously as stored-program computers. Many programs thus may be running at one time or a combination of processors can be involved in one problem which may require a variety of input/output tasks as well as use of the central memory and the central processor(s).

The peripheral processors act as system control computers and input/output processors. This permits the central processor to continue computation while the peripheral processors do the slower input/output and supervisory operations.

Each processor has a 12-bit, 4096 word random-access memory (independent of central memory) with a cycle time of 1000 ns. Execution time of processor instructions is dependent on memory cycle time.

INPUT/OUTPUT

All processors communicate with external equipment and each other via the independent, bidirectional I/O channels. The number of channels depends on the number of peripheral processors in the system. All channels are 12-bit (plus control) and each may be connected to one or more external devices. Only one external equipment can utilize a channel at one time, but all channels can be simultaneously active. Data is transferred into or out of the system in 12-bit words; each channel has a single register which holds the data word being transferred in or out. Each channel operates at a maximum rate of one word per microsecond.

Data flows between a peripheral processor memory and the external device in blocks of words (a block may be as small as one word). A single word may be transferred between an external device and the A register of a peripheral processor.

The I/O instructions direct all activity with external equipment. These instructions determine the status of, and select an external device on any channel and transfer data to or from the selected device. Two channel conditions are made available to all processors as an aid to orderly use of channels.

- Each channel has an active/inactive flag to signal that it has been selected for use and is busy with an external device.
- Each channel has a full/empty flag to signal that a word (function or data) is available in the register associated with the channel.

Either state of both flags can be sensed. In general, an I/O operation involves the following steps:

1. Determine channel inactive
2. Determine equipment ready
3. Select equipment
4. Activate channel
5. Input/Output data
6. Disconnect channel

One peripheral processor may communicate with any other over any channel which has been selected for output by one and for input by the other. A common channel can be reserved for interprocessor communication and for preservation of order by keeping track of equipment and channel status.

REALTIME CLOCK

A real-time clock reading is available on a channel which is not counted as a regular channel. The clock period is 4096 major cycles. The clock starts with power on and runs continuously. It cannot be preset or altered. The clock may be used to determine program running time or other functions such as time-of-day, as required.

CENTRAL MEMORY COMMUNICATIONS

Each processor exchanges data with central memory in blocks of words. Five successive 12-bit processor words are assembled into a 60-bit word and sent to central memory for a Write operation. A 60-bit central memory word is disassembled into five 12-bit words and sent to successive locations in a processor memory for a Read operation. Separate assembly

(write) and disassembly (read) paths to central memory are shared by up to 10 peripheral processors. Up to four processors may be writing in central memory while another four are simultaneously reading from central memory. Systems with more than 10 peripheral processors have another set of read and write paths.

PERIPHERAL PROCESSOR SYSTEM RELATIONSHIPS

The peripheral processors generally are not used to solve complex arithmetic and logical problems. Usually they are used to perform I/O operations for running central processor programs and for organizing data (operands, addresses, constants, program length, relative starting address, exit mode), to store in central memory.

THE EXCHANGE JUMP

An Exchange Jump instruction starts (or interrupts) the central processor and provides the central processor with the starting address of a problem stored in central memory. The central processor, at the next convenient breakpoint, then exchanges the contents of its A, B, and X registers, its program address, relative starting address, length of program, Exit mode and Extended Core Storage parameters with the stored information for the new program. A later Exchange Jump would be needed to call for a return to the incomplete interrupted program.

INTERLOCK REGISTER AND ACCESS CHANNEL

This is a 64- or 128-bit flag register with a special access channel (15_8). Each access channel accommodates up to 10 peripheral processors so if the system has more than 10 processors, a second access channel is utilized. The interlock register provides a means for all peripheral processors to communicate with each other without the necessity for making central memory references. The peripheral processors can perform set, clear, test, and read operations on the interlock register.

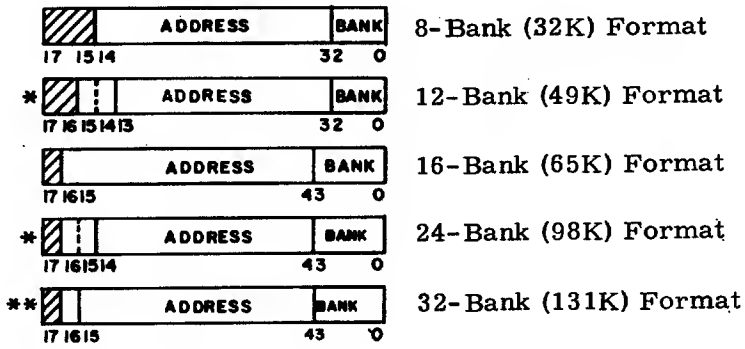
The access channel has a 12-bit input register and a 12-bit output register. The channel assumes a Full status whenever one peripheral processor does an output (to prevent any other peripheral processor from interrupting. The Full status is cleared only by the concerned peripheral processor doing an input. The access channel is designated as channel 15.

CENTRAL MEMORY

Central memory is a core memory with a capacity of 32K, 49K, 65K, 98K, or 131K 60-bit words in 8, 12, 16, 24, or 32 banks of 4096 words each. The banks are logically independent and may be phased into operation at 100 nsec intervals. The central memory address and data control mechanisms permit a word to move to or from central memory every 100 nsec. (32K and 49K memories pause 200 nsec after every eight words.) Addresses, written or compiled in conventional manner, reference consecutive banks and thus make efficient use of the bank phasing technique.

ADDRESS FORMATS

The location of each word in central memory is identified by an assigned address, which consists of 18 bits. Address formats are shown below for 8-bank (32K), 12-bank (49K), 16-bank (65K), 24-bank (98K), and 32-bank (131K) systems. Within the address format, the bank portion specifies one of 12, 24, or 32 banks; 12-bit address defines one of the 4096 separate locations within the specified bank.



ACCESS

References to central memory from all areas of the system (central processor and peripheral processors) and extended core storage go to a common memory control and are issued to all banks in central memory. The control accepts addresses from the various sources under a priority system and at a maximum rate of one address every minor cycle.

An address is sent to all memory banks. The correct bank, if free (the bank ignores the address if it is busy processing a previous address), accepts the address and indicates this to the memory control. The associated data word is then sent to or stored from a central data distributor. The memory control issues addresses at a maximum rate of one every 100 nsec.

*One bit of bank portion is supplied by address bit 2^{15} or 2^{14} (49K) or $2^{16} + 2^{15}$ (98K), depending on the Section/Chassis configuration.

** Bit-16 0 = bank 00_8 - 17_8 address 000000_8 - 177777_8
1 = bank 20_8 - 37_8 address 200000_8 - 377777_8

The memory control saves, in a hopper mechanism, each address that it sends to central memory and then reissues it (and again saves it) under priority control in the event that it is not accepted because of bank conflict. The address issue-save process repeats until the address is accepted, at which time the address is dropped from the hopper and the read or store data word is distributed. A fixed time lapse from address-issue to the memory-accept synchronizes the action taken.

The previously unaccepted address has highest priority among addresses to central memory. The central processor and peripheral processors (all share a common path to the memory control) follow in priority.

A data distributor, which is common to all processors, handles all data words to and from central memory. Up to 10 peripheral processors share one read path and one write path to the distributor. A series of buffer registers in the distributor provides temporary storage for words to be written into storage when the addresses are not immediately accepted because of bank conflict. Systems with more than 10 peripheral processors have another set of read and write paths.

Each group of four banks communicates with the distributor on separate 60-bit read and write paths, but only one word moves on the data paths at one time. However, words can move at 100 nsec intervals between the distributor and central memory or distributor and address-sender.

Data words and addresses are correlated by control information tags entered in the memory control with the address. The tags identify the address sender, origin/destination of data, and whether the address is a Read, Write, or Exchange Jump address.

MEMORY PROTECTION

All central processor references to central memory for new instructions, or to read and store data, are made relative to the Reference Address. The Reference Address defines the lower limit of a central memory program. Changes to the Reference Address permit easy relocation of programs in central memory.

During an Exchange Jump, an 18-bit Reference Address and an 18-bit Field Length (parts of the Exchange Jump package) are loaded into their respective registers to define the central memory limits of the program initiated by the Exchange Jump.

The relationship between absolute memory address, relative memory address, Reference Address (RA), and Field Length (FL) is indicated in Figure 1-2.

The following relationships must be true if the program is to operate within its bounds:

$RA \leq (RA + P) < (RA + FL)$ (Absolute Memory Addresses), or

$0 \leq P < FL$ (Relative Memory Addresses)

NOTE

FL is the number of 60-bit words in the program. It is not an address.

To avoid possible "artificial" range faults, instructions should not be stored at absolute address [(RA + FL) - 1] because an instruction produces a range fault when the (look-ahead) Read Next Instruction occurs to (RA + FL). Data rather than instructions should always be stored in addresses near absolute location (RA + FL).

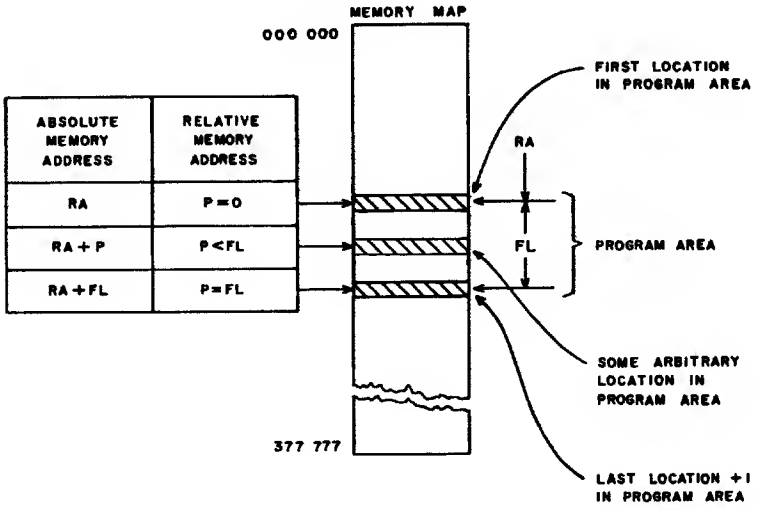


Figure 1-2. Memory Map

An optional exit condition (EM in the Exchange Jump package) allows the central processor to stop on a memory reference outside the limits expressed above.

PROGRAMMING INFORMATION 2.

CENTRAL PROCESSOR PROGRAMMING

Central processor program instructions are stored in central memory. Each 60-bit memory location may hold four 15-bit instructions, two 30-bit instructions or a combination of 15 and 30-bit instructions.

The central processor reads 60-bit words from central memory and stores them in an instruction stack which is capable of holding up to eight 60-bit words. These programming instructions refer only to CPU0 in dual CPU systems. Programming for the unified arithmetic unit (CPU1) is described in the Model 73 reference manual volume 1, publication number 60347100.

Each instruction is sent in turn to a series of instruction registers for interpretation and testing and is then issued to the arithmetic unit for execution. The arithmetic unit obtains the instruction operands from, and stores results in, the 24 operating registers. The reservation control records busy operating registers to avoid conflicts and to ensure that the original instructions do not get out of order.

INSTRUCTION FORMATS

Groups of bits in an instruction are identified by the letters f, m, i, j, k, and K as shown in Figure 2-1. All letters represent octal digits except K, which represents an 18-bit constant. The f and m digits are the operation code and identify the type of instruction. In a few instructions the i designator becomes a part of the operation code.

In most 15-bit instructions, the i, j, and k digits each specify one of the eight operating registers where operands are found and where the results of the operation are to be stored. In other 15-bit instructions, the j and k digits provide a 6-bit shift count.

In 30-bit instructions, the i and j digits each specify one of the eight operating registers where one operand is found and where the result is to be stored, and K is taken directly as an 18-bit second operand.

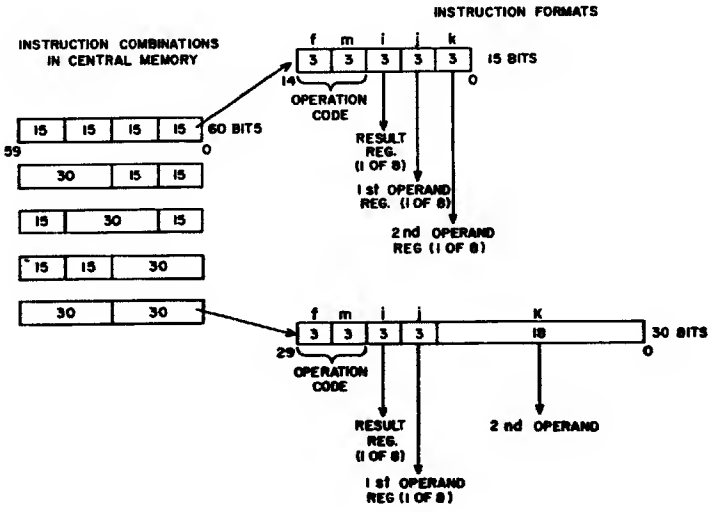


Figure 2-1. Central Processor Instruction Formats

OPERATING REGISTERS

An Exchange Jump instruction from a peripheral processor enters initial values in the operating registers to start central processor operation. Subsequent address modification instructions provide the addresses required to retrieve and store data.

In order to provide a compact symbolic language, the 24 operating registers are identified by letters and numbers:

A = address register (A0, A1 ... A7)
B = increment register (B0, B1 ... B7)
X = operand register (X0, X1 ... X7)

X REGISTERS

The operand registers hold operands and results. Five registers (X1 - X5) hold read operands from central memory, and two registers (X6 - X7) hold results to be sent to central memory (Figure 2-2). Operands and results transfer between memory and these registers as a result of placing a quantity into a corresponding address register (A1 - A7).

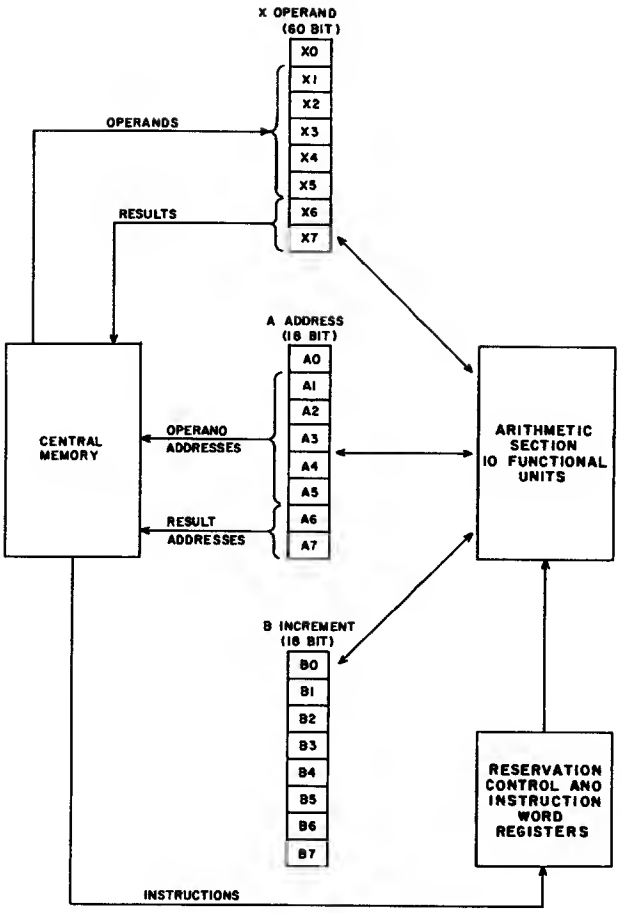


Figure 2-2. Central Processor Operating Registers

Placing a quantity into an address register A1 - A5 produces an immediate memory reference to that address and reads the operand into the corresponding operand register X1 - X5. Similarly, placing a quantity into address register A6 or A7 stores the word in the corresponding X6 or X7 operand register in the new address.

A REGISTERS

An increment instruction places a result in address register Ai (where "i" = 0-7) in any one of three ways:

- By adding an 18-bit signed constant K to the contents of any A, B, or X register.
- By adding the contents of any B register to any A, B, or X register.
- By subtracting the contents of any B register from any A register or any other B register.

The A0 and X0 registers are independent and have no connection with central memory. They may be used for scratch pad or intermediate results. Note the special use of A0 and X0 when executing extended core storage communication instructions.

B REGISTERS

The B registers have no connection with central memory. The B0 register is fixed to provide a constant zero (18-bit) which is useful for various tests against zero, providing an unconditional jump modifier, etc. In general, the B registers offer means for program indexing. For example, B4 may store the number of times a program loop has been traversed, thereby providing a terminating condition for a program exit.

PROGRAM ADDRESS REGISTER

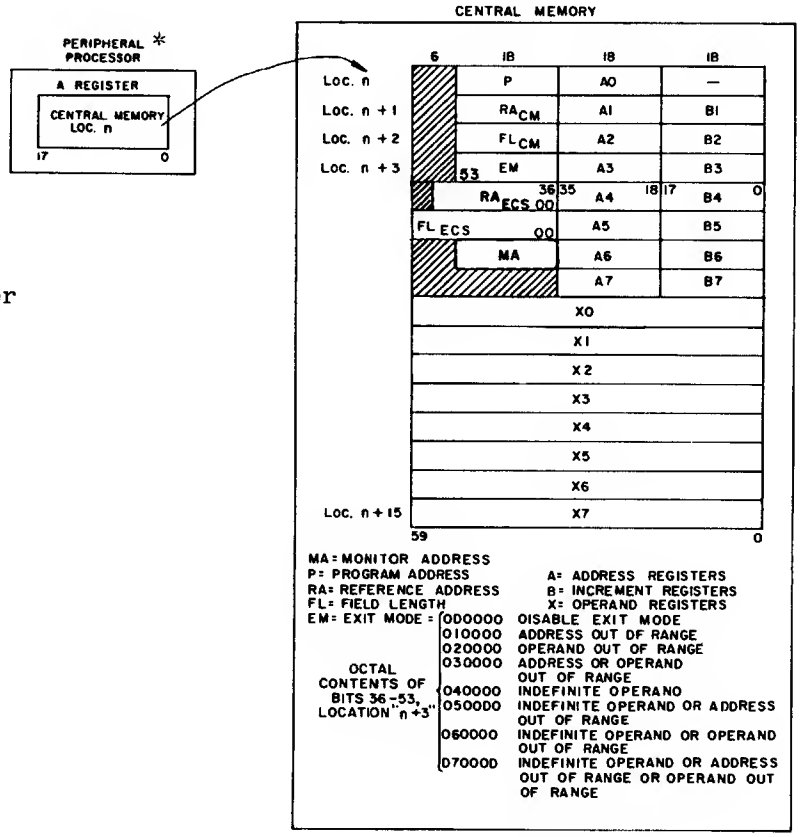
An 18-bit P register serves as a program address counter and holds the address for each program step. P is advanced to the next program step in the following ways:

1. P is advanced by one when all instructions in a 60-bit word have been extracted and sent to the instruction registers.
2. P is set to the address specified by a Go To ... (branch) instruction. If the instruction is a Return Jump, (P) + 1 is stored before the branch to allow a return to the sequence after the branch. Branch instructions to a new program start the program with the instruction located in the highest order position of the 60-bit word.
3. P is set to the address specified in the Exchange Jump package.

EXCHANGE JUMP

An Exchange Jump instruction starts or interrupts the central processor and provides central memory with the first address of a 16-word package in central memory. The Exchange Jump package (Figure 2-3) provides the following information on a program to be executed.

1. Program address (P)
2. Reference Address for Central Memory (RA_{CM})
3. Field length of program for Central Memory (FL_{CM})
4. Reference Address for Extended Core Storage (RA_{ECS})
5. Field length of program for Extended Core Storage (FL_{ECS})
6. Program exit mode (EM)
7. Initial contents of the eight A registers
8. Initial contents of the eight X registers
9. Initial contents of B registers B1 - B7 (B0 is fixed at 0)
10. Monitor Address (MA)



*The Central Processor may also initiate an exchange jump.

Figure 2-3. Exchange Jump Package

The central processor enters the information about a new program into the appropriate registers and stores the corresponding and current information from the interrupted program at the same 16 locations in central memory. Hence, the controlling information for two programs is exchanged. A later Exchange Jump may return an interrupted program to the central processor for completion. The normal operation of the A and X register is not active during the Exchange Jump so that the new entries in A are not reflected changes in X.

NOTE

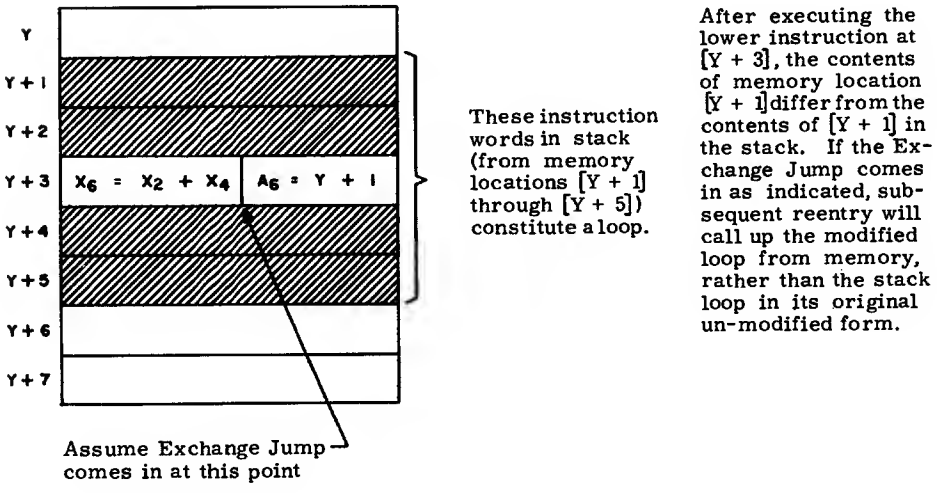
When an Exchange Jump interrupts the central processor, several steps occur to ensure leaving the interrupted program in a usable state for re-entry:

1. Issue of instructions halts after issuing all instructions from the current instruction word in the instruction stack.
2. The Program Address register, P, is set to the address of the next instruction word to be executed.
3. The issued instructions are executed, and then:
4. The parameters for the two programs are exchanged.

A subsequent Exchange Jump can then re-enter the interrupted program at the point at which it was interrupted, with no loss of program continuity.

To preserve the integrity of an "in-stack" loop (in the event of an Exchange Jump), it is illegal to modify the contents of any memory address which holds an executable instruction (or instruction word) contained within the loop.

EXAMPLE:



REFERENCE ADDRESS

All central processor references to central memory, whether for new instructions, or to fetch and store data, are made relative to the Reference Address. This allows easy relocation of a program in central memory. The Reference Address or beginning address allows the central processor to stop on a memory reference outside these limits.

The Program Address register, P, defines the location of a program step within the limits prescribed. Each reference to memory to fetch instructions is made to the address specified by P + RA. The program relocation is thus conveniently handled through a single change to RA. A P = 0 condition specifies address zero and hence RA. This address is reserved for recording program exit (error) conditions and should not be used to store data or instructions of a program.

EXIT MODE

The Exit mode allows the programmer to select Exit or Stop conditions for the central processor. Exit selections (EM) are loaded into bits 36-53 of memory location "n+3" of the Exchange Jump package (Figure 2-3). When the Exchange Jump to that package occurs, the exit selections are stored in the central processor and the exit occurs as soon as the selected condition is sensed.

NOTE

The CEJ/MEJ switch permits selection of a non-stop mode at its ENABLE position. In this mode, any stop condition is treated as an instruction to jump to the monitor address in the Exchange Jump package if the moniter flag is clear. If the monitor flag is set (CPU in moniter mode) the CPU will stop. If the CEJ/MEJ switch is at DISABLE position, the stop condition will cause a stop.

EXCHANGE JUMP PACKAGE

The Exit selections are stored in bits 36-53 (EM) of address "n+3" in the Exchange Jump package. The significance of the bits is shown here octally with the binary equivalent shown in the explanations:

EM = 000000 Disable Exit mode - no Exit selections made.

= 010000 Address out of range - (Bit 48 set)

a. an attempt to reference either central memory or extended core storage outside established limits, or

b. the word count, [(Bj) + K], of an extended core storage Communication instruction is negative.

(For details on action when an address is out of range, refer to the Increment and extended core storage instruction descriptions.)

= 020000 Operand out of range - floating point using an infinite operand (see Range Definitions under Floating Point Arithmetic). (Bit 49 set)

= 030000 Address or operand out of range. (Bits 48 and 49 set)

= 040000 Indefinite operand - floating point arithmetic sequence attempted to use an indefinite operand (see Range Definitions). (Bit 50 set)

= 050000 Indefinite operand or address out of range. (Bits 48 and 50 set)

= 060000 Indefinite operand or operand out of range (infinite operand). (Bits 49 and 50 set)

= 070000 Indefinite operand or operand or address out of range. (Bits 48, 49, and 50 set)

ERROR EXIT STATUS BITS

The Error Exit Status bits are stored in EM bits 51 and 52. These bits are stored during an Exchange Jump and result from an error condition. The octal formats and descriptions are:

EM = 1X0000 Error condition detected and exchanged out. [Error Exit not executed.] (Bit 51 set)

EM = 2X0000 Error Exit completed, halt central processor. [P = zero] (Bit 52 set)

The Error Exit Status bits 51 and 52 are for hardware action only, and should not be set via software. If EM 51 or 52 is set by software, erroneous error conditions will be reported.

If EM bit 51 is set in an Input Exchange Package the CPU will Error Exit (no mode bit will be set in RA). If EM bit 52 is set in an Input Exchange Package the CPU will Error CEJ if the CEJ/MEJ hardware switch is enabled; the CPU will halt if the switch is disabled.

EM 52 is set by the hardware to prevent execution of a bad program; if re-execution is desired, EM 52 must be cleared by the software via the operating system.

RESULTS OF EXIT

The Reference Address (RA) for any program is typically left cleared to all zeros. When an error exit is taken, the central processor records at RAcm the exit condition (bits 48-53) and the Program Address (bits 30-47) at exit time (refer to the format below).

NOTE

The Exit condition(s) recorded at RA are all the Exit conditions detected since the last Exchange Jump, regardless of whether or not they were selected. Thus, combinations of error Exit conditions (03, 05, 06 or 07) can appear at RA:

a. When at least one Exit condition was selected and the selected condition plus another condition occurred since the last Exchange Jump, or

b. When more than one Exit condition was selected and each occurred in the same minor cycle.

(P)=P,P+1, OR P+2 AT TIME OF ERROR EXIT.

For error stops, RAcm (P) gives only an approximate location of the error since the central processor may have issued other instructions (one of which may have been a branch) before the error condition was sensed.

ADDRESS OUT OF RANGE

On an Address Out of Range, hardware action differs from that previously outlined. In some cases, a stop occurs when an address is out of range even though an Exit mode stop is not selected for this condition. Table 2-1 summarizes hardware action for operations which reference addresses that are out of range. Floating point arithmetic is discussed separately.

TABLE 2-1. EXIT MODE: ADDRESS OUT OF RANGE

	Hardware Action	
Operation	Exit Mode Selected	Exit Mode Not Selected
RNI to an address that is out-of-range (occurs when an instr. is located in absolute address (RA + FL) - 1).	1. Detect error condition 2. Write EM and (FL-1) into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7).	1. Detect error condition 2. Stop by reading (AAZ) (Absolute Address Zero) 3. Nothing stored in RA 4. (P) = out of range P or (FL)
Branch to an address that is out-of-range.	1. Detect error condition 2. Write EM and jump address in RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)	1. Detect error condition 2. Stop by reading (AAZ) 3. Nothing stored in RA 4. (P) = jump address
Read Operand (Increment)	1. Detect error condition 2. Write EM and (P) + 1 or (P) + 2 into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)	1. Detect error condition 2. Store zeros in X_i. 3. Nothing stored in RA 4. Continue program
Write Operand (Increment)	1. Detect error condition 2. Write EM and (P) + 1 or (P) +2 into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)	1. Detect error condition 2. X_i not stored; (X_i) unchanged, and A_i = Increment Result 3. Continue program.

UNCONDITIONAL EXIT

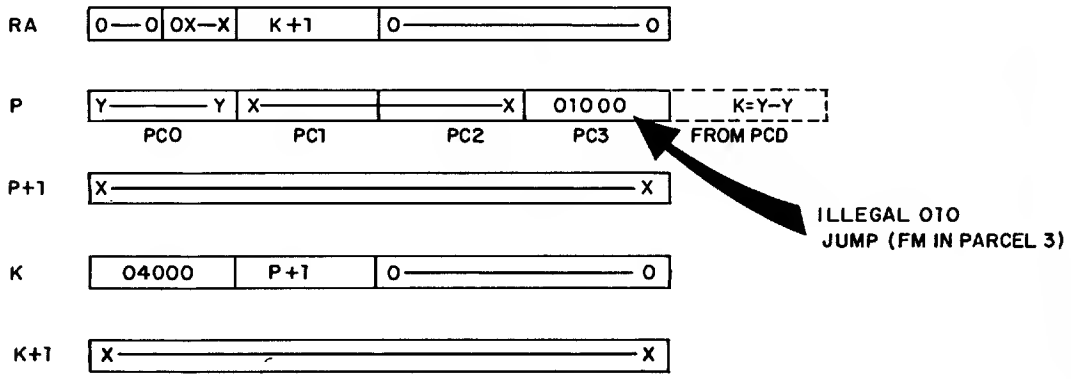
All central processor attempts to execute an illegal or non-available instruction will force an Error Exit. There are no mode selections for these conditions. Table 2-1.1 lists the hardware action for the different types of illegal instructions:

TABLE 2-1.1. UNCONDITIONAL EXIT ACTIONS

Operation	Hardware Action
Illegal 30 bit Instruction (fm of 30 bit instruction in parcel 3)	1. Detect error condition 2. Store P+1 or P+2 into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)
Illegal 01X instruction (fm of 01X instruction not in parcel 0). Does not include 010 jump (see text).	1. Detect error condition 2. Store P+1 or P+2 into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)
Illegal ECS instruction (execution of ECS instruction without ECS)	1. Detect error conditon 2. Store P into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)
Illegal CEJ instruction (execution of CEJ instruction when the CEJ/MEJ switch is disabled)	1. Detect error condition 2. Store P into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)
Illegal CMI instruction 464, 465, 466, or 467. (CMI instruction not available in CYBER Model 74.)	1. Detect error condition 2. Store P+1 or P+2 into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)
Illegal stop instruction, fm=00 (non-stop if the CEJ/MEJ switch is enabled.)	1. Detect error condition 2. Store P+1 or P+2 into RA 3. Clear P (RNI RA) 4. Error CEJ is executed by reading RA. (Refer to note on page 2-7.)

A unique condition exists in the storing of RA when an Illegal (fm in parcel 3) 010 Return Jump to K is executed. The contents of RA will be the K address + 1 (K + 1) specified by the 010 Jump. (Note in the example that K is taken from parcel 0.) The address K will automatically contain an unconditional 04 Jump to K if (Bi) = (Bj) to the current Addres + 1 (P + 1) of the Illegal 010 Jump instruction.

EXAMPLE:



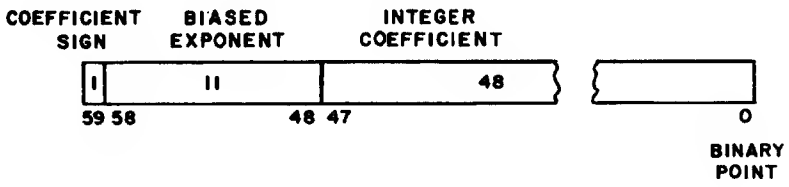
FLOATING POINT ARITHMETIC

FLOATING POINT ARITHMETIC THEORY

Floating point arithmetic takes advantage of the ability to express a number with the general expression kB^n, where:

k = coefficient
B = base number
n = exponent, or power to which the base number is raised

The base number is constant (2) for binary-coded quantities and is not included in the general format. The 60-bit floating-point format is shown below. The binary point is considered to be to the right of the coefficient, thereby providing a 48-bit integer coefficient, the equivalent of about 14 decimal digits. The sign of the coefficient is carried in the highest order bit of the packed word. Negative numbers are represented in one's complement notation.



The 11-bit exponent carries a bias of 2^{10} (2000_8) when packed in the floating point word (biased exponent sometimes referred to as: "characteristic"). The bias is removed when the word is unpacked for computation and restored when a word is packed into floating format. Table 2-2 lists (in decimal and octal notation) the complete range of permissible exponents and the octal form of the corresponding positive and negative floating point words.

Thus, a number with an exponent of 342 would appear as 2342_8; a number with an exponent of -160 would appear at 1617_8. Exponent arithmetic is done in ones' complement notation. Floating point numbers can be compared for equality and threshold.

TABLE 2-2. RANGE OF PERMISSIBLE EXPONENTS

Exponent (n)			Representation of kxB^n (Octal)	
Decimal	Octal		Positive Coefficient	Negative Coefficient
+1023	+1777	(infinite operand)	3777 X X	4000 X X
+1022	+1776		3776 X X	4001 X X
.	.		.	.
.	.		.	.
.	.		.	.
.	.		.	.
+1	+1		2001 X X	5776 X X
+0	+0		2000 X X	5777 X X
-0	-0	(indefinite operand)	1777 X X	6000 X X
-1	-1		1776 X X	6001 X X
.	.		.	.
.	.		.	.
.	.		.	.
.	.		.	.
-1023	-1777		0000 X X	7777 X X

NORMALIZING

Normalizing a floating point quantity shifts the coefficient left until the most significant bit is in bit 47. Sign bits are entered in the low-order bits of the coefficient as it is normalized. Each shift decreases the exponent by one. Two normalized input operands cannot be used during an integer multiply operation because they will be treated as floating point operands and will cause the storage of the underflow results. Therefore, normalizing floating point quantities used as operands should be done whenever there is danger that they may be interpreted as integers.

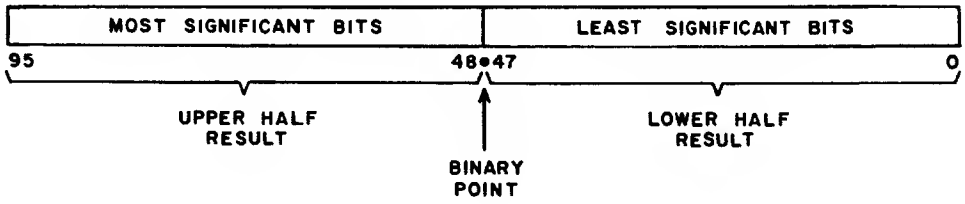
ROUNDING

A round bit is added (optionally) to the coefficient during an arithmetic process and has the effect of increasing the absolute value of the operand or result by one-half the value of the least significant bit. Normalizing and rounding are not automatic during pack or unpack operations so that operands and results may not be normalized.

SINGLE AND DOUBLE PRECISION

The floating point arithmetic instructions generate double-precision results. Use of unrounded operations allows separate recovery of upper and lower half results with proper exponents; only upper half results can be obtained with rounded operations.

Double precision results appear as follows:



RANGE DEFINITIONS

A result with an exponent so large that it exceeds the upper limit of octal 3777 (overflow case) is treated as an infinite quantity. A coefficient of all zeros and an exponent of octal 3777 or 4000 is packed for this case. An optional exit is provided when an attempt is made to use an infinite operand in floating arithmetic sequences since its use may propagate an indefinite result as shown in Table 2-3. No error exit occurs when an infinite or indefinite result is generated in a sequence.

TABLE 2-3. INDEFINITE FORMS

$\infty - \infty$	= INDEFINITE	$\infty + N = \infty$
$\infty + \infty$	= INDEFINITE	$\infty + N = \infty$
$\infty \bullet 0$	= INDEFINITE	$\infty - N = \infty$
$0 + 0$	= INDEFINITE	$N \div 0 = \infty$
INDEFINITE +, −, +, • (X)	= INDEFINITE	$0 + \infty = 0$
$\infty + \infty$	= ∞	$0 \bullet 0 = 0$
$\infty \bullet \infty$	= ∞	$0 \div N = 0$
$\infty + 0$	= ∞	$N \div \infty = 0$

WHERE: ∞ = INFINITY, N = INTEGER, X = ∞, N OR 0.

A resulting exponent which is less than the lower limit of octal 0000 (underflow case) is treated as a zero quantity. This quantity is packed with a zero exponent and zero coefficient. No exit is provided for underflow. A partial underflow result with an exponent of octal 0000 and a coefficient which is not zero is a non-zero quantity and is packed with a zero exponent and the non-zero coefficient. A precaution must be taken to normalize when using partial underflow results as operands in subsequent floating point multiply operations. This will prevent these operands from being interpreted as integer operands resulting in an integer multiply operation.

Use of either infinity or zero as operands may produce an indefinite result. An exponent of octal 1777 and a zero coefficient are packed in this case, and an optional exit provided. In the special case of integer multiply, both operands have zero coefficients and no packing and no exit take place. Note that zero, infinite, and indefinite results are generated or regenerated in floating arithmetic operations only. The branch instructions test for infinite or indefinite quantities.

In all floating arithmetic operations, an attempt to normalize an indefinite quantity returns the original quantity, e.g., if the number 17770237 ... were to be normalized, the result would be the same as the original number. Exit mode can be made to occur on detecting an indefinite quantity.

Exit mode tests for infinite and indefinite operands are made in the shift (normalize), floating add, multiply, and divide sequences. The 12 most significant bits of each operand are tested for these special forms.

In the multiply and divide sequences (but not in a floating add) there is a special test for zero operands as determined by the 12 most significant bits.

Thus, the special operand forms (in octal) are:

3777X...X	(+∞)	infinite operands
4000X...X	(-∞)	
1777X...X	(+IND)	indefinite operands
6000X...X	(-IND)	
0000X...X	(+0)	zero operands for Multiply and Divide (If both operands have +0 or -0 exponents, integer multiply results)
7777X...X	(-0)	

Whenever infinite, indefinite, or zero results are generated in accordance with the rules given in Table 2-3 and only the following octal words can occur as results:

37770...0	= +∞	(result)
40000...0	= -∞	(result)
17770...0	= +IND	(result)
00000...0	= +0	(result)

Note that in these cases the 48 least significant bits of the result are zeros. Indefinite and zero results generated in accordance with Table 2-3 are always positive, but the sign of infinite results is determined by the usual algebraic sign convention. For example:

(+0) / (-0)	= +IND	= 17770...0
(+N) * (-0)	= +0	= 00000...0
(-∞) / (-0)	= +∞	= 37770...0
(+∞) / (-0)	= -∞	= 40000...0

There is no special treatment of zero operands in floating add operations. Zero coefficients and the forms 0000X...X and 7777X...X are not specially detected, and unstandardized zero results can be produced. (See description of 30 instruction).

OVERFLOW AND UNDERFLOW

Exponents lying outside the range -1777_8 to $+1777_8$ cannot be generated during execution of a floating point arithmetic instruction or during execution of a Normalize instruction. An attempt to generate an exponent greater than $+1777_8$ yields an infinite result (overflow case). An attempt to generate an exponent less than -1777_8 yields a zero result (underflow case). All cases of overflow and underflow are listed in Table 2-4.

CONVERTING INTEGERS TO FLOATING FORMAT

Conversion of integers to floating point format makes use of the shift sequence and the zero constant in increment register B0. The B0 quantity provides for generation of exponent bias in this case. For example, the instructions:

- Sum of Bj and Bk to Xi (where i = 2, j = 3, k = 4)
- Pack Xi from Xk and Bj (where i = 2, j = 0, k = 2)

form an 18-bit signed integer in operand register X2 as a result of the addition of the contents of increment register B3 and B4. The integer coefficient with its sign, plus the octal 2000 exponent is then packed into the floating format shown earlier. The coefficient is not normalized; normalizing may be accomplished with a Normalize instruction.

TABLE 2-4. OVERFLOW AND UNDERFLOW CONDITIONS

Overflow		
Instructions	Overflow Condition	Result
Normalize (24, 25)	None	---
Upper Sum (30, 31, 34, 35)	None (see Note 1)	---
Lower Sum (32, 33)	None	---
Upper Product (40, 41)	$*n_1 + n_2 + 60_8 \geq 2000_8$	X_i = 3777 0....0_8 or 4000 0....0_8 (True Sign)
Lower Product (42)	$n_1 + n_2 \geq 2000_8$	
Quotient (44, 45)	$n_1 - n_2 - 57_8 \geq 2000_8$	

Underflow		
Instructions	Underflow Condition	Result
Normalize (24 only)	Initial coefficient = ±0	X_i = 0000 0....0_8, (Bj) = 60_8
Normalize (24, 25)	Final Exponent $\leq -2000_8$	X_i = 0000 0....0_8, (Bj) are correct. (See Note 2.)
Upper Sum (30, 31, 34, 35)	None	---
Lower Sum (32, 33)	Final Exponent $\leq -2000_8$	X_i = 0000 0....0_8
Upper Product (40, 41)	$n_1 + n_2 + 57_8 \leq -2000_8$	X_i = 0000 0....0_8
Lower Product (42)	$n_1 + n_2 - 1 \leq -2000_8$	
Quotient (44, 45)	$n_1 - n_2 - 60_8 \leq -2000_8$	

*N_1 and n_2 are the initial exponents.

Note 1. Overflow of Upper Sum: Overflow cannot occur unless one operand is infinite. In this case the result is as indicated. If a one-place Right Shift occurs when the larger operand exponent is equal to $+1776_8$, a correct result with exponent $+1777_8$ is generated.

Note 2. Underflow of Exponent During Normalization: The final (Bj) are the same as if underflow had not occurred. In particular, if the initial coefficient is zero, (Bj) are equal to 60_8.

FLOATING POINT ARITHMETIC TABLES

The following is a tabulation of operations (Add, Subtract, Multiply, Divide) using various combinations of operands to supplement Table 2-1. The key to operands and results used is as follows:

KEY:

Operands			Results		
+0	=	0000 X...X	0	=	0000 0...0
-0	=	7777 X...X	IND	=	1777 0...0
+∞	=	3777 X...X	INT	=	Integer result
-∞	=	4000 X...X	+∞	=	3777 0...0
+IND	=	1777 X...X	-∞	=	4000 0...0
+INT	=	Integer number	Δ	=	Any result except 0,
-INT	=	Integer number			IND, or ±∞
W	=	Any word except ±∞, ± IND	□	=	Any result except IND or ±∞
N	=	Any word except ±∞, ± IND, or ±0			

ADD (INSTRUCTIONS 30, 32, 34)

Xi=Xj+Xk

	Xk			
Xj	W	+ ∞	- ∞	± IND
W	□	+ ∞	- ∞	IND
+ ∞	+∞	+ ∞	IND	IND
- ∞	-∞	IND	- ∞	IND
± IND	IND	IND	IND	IND

SUBTRACT (INSTRUCTIONS 31, 33, 35)

Xi=Xj-Xk

	Xk			
Xj	W	+ ∞	- ∞	± IND
W	□	- ∞	+ ∞	IND
+ ∞	+ ∞	IND	+ ∞	IND
- ∞	- ∞	- ∞	IND	IND
± IND	IND	IND	IND	IND

MULTIPLY (INSTRUCTIONS 40, 41, 42)

Xi=Xj*Xk

Xj \ Xk	+N	-N	+0	-0	+∞	-∞	± IND
+N	Δ	Δ	0	0	+∞	-∞	IND
-N	Δ	Δ	0	0	-∞	+∞	IND
+0	0	0	INTEGER MULTIPLY (NOTE)		IND	IND	IND
-0	0	0			IND	IND	IND
+∞	+∞	-∞	IND	IND	+∞	-∞	IND
-∞	-∞	+∞	IND	IND	-∞	+∞	IND
± IND	IND	IND	IND	IND	IND	IND	IND

NOTE: If both operands are normalized and the exponents are zero, positive underflow results are reported.

DIVIDE (INSTRUCTIONS 44, 45)

Xi=Xj/Xk

Xj \ Xk	+N	-N	+0	-0	+∞	-∞	± IND
+N	Δ	Δ	+∞	-∞	0	0	IND
-N	Δ	Δ	-∞	+∞	0	0	IND
+0	0	0	IND	IND	0	0	IND
-0	0	0	IND	IND	0	0	IND
+∞	+∞	-∞	+∞	-∞	IND	IND	IND
-∞	-∞	+∞	-∞	+∞	IND	IND	IND
± IND	IND	IND	IND	IND	IND	IND	IND

SHORT WORD INTEGER MULTIPLICATION TABLES

KEY:

Operands			Results		
+0	= 0000	0...0	+0	= 0000	0...0
-0	= 7777	7...7	+0	= 0000	0...0
+INT	= 0000	X...X	+INT	= 0000	X...X (NOTE)
-INT	= 7777	X...X	-INT	= 7777	X...X (NOTE)

NOTE: Unless both operands are normalized, in which case complete positive underflow results.

INTEGER MULTIPLY (INSTRUCTION 42)

Xi=Xj*Xk

Xj	Xk			
	+INT	-INT	+0	-0
+INT	+INT	-INT	+0	-0
-INT	-INT	+INT	-0	+0
+0	+0	-0	+0	-0
-0	-0	+0	-0	+0

FIXED POINT ARITHMETIC

Fixed point addition and subtraction of 60-bit numbers is handled in the long add functional unit section. Negative numbers are represented in one's complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 59) and the binary point is at the right of the low-order bit position (bit 0).

The increment functional units provide an 18-bit fixed point add and subtract facility. Negative numbers are represented in one's complement notation and overflows are ignored. The sign bit is in the high-order bit position (bit 17), and the binary point is at the right of the low-order bit position (bit 0).

Integer multiplication is handled as a subset operation of the Floating Multiply (42) instruction. The integer multiply requires that both of the 47-bit integer operands have zero exponents and one or both operands are not normalized. The result is 48 bits with sign extension. Both operands normalized cause positive underflow results to be reported. If the results exceed 48 bits, overflow will not be detected. (See 40 instruction for overflow detection.)

An integer divide takes several steps. For example, an integer quotient X1 = X2/X3 is produced by the following steps:

	Instructions	Remarks
1.	Pack X2 from X2 and B0	Pack X2
2.	Pack X3 from X3 and B0	Pack X3
3.	Normalize X3 in X0 and B0	Normalize X3 (divisor)
4.	Floating quotient of X2 and X0 to Xi	Divide
5.	Unpack X1 to X1 and B7	Unpack quotient
6.	Shift X1 nominally left B7 places	Shift to integer position

The divide requires that:

1. both integer (2^{47} maximum) operands be in floating format

and 2. the divisor be shifted 48 places left

or 3. The quotient be shifted 48 places right

or 4. any combination of n left-shifts of the divisor and 48-n right shifts of the quotient be accomplished.

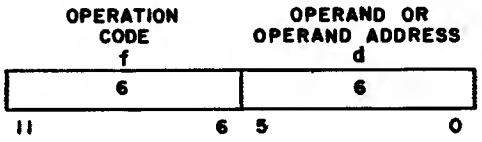
The Normalize X3 instruction shifts the divisor n places left ($n \geq 0$), providing divisor exponent of -n. The quotient exponent then is: $0 - (-n) - 48 = n - 48 \leq 0$.

After unpacking and shifting nominally left, the negative (or zero) value in B7 shifts the quotient 48 - n places right, producing an integer quotient in X1. A remainder may be obtained by an integer multiply of X1 and X3 and subtracting the result from X2.

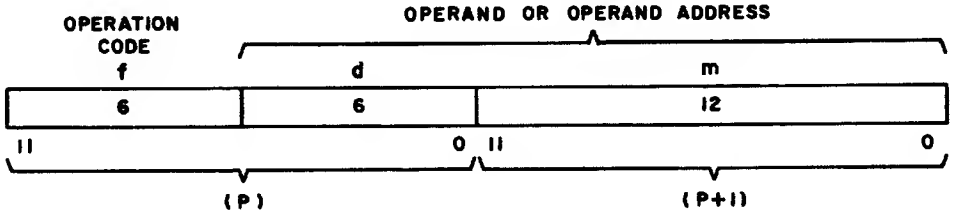
PERIPHERAL PROCESSOR PROGRAMMING

INSTRUCTION FORMATS

Peripheral processor instructions are either in a 12-bit or a 24-bit format. The 12-bit format has a 6-bit operation code designated f and a 6-bit operand or operand address designated d. The formats are made up as follows:



The 24-bit format uses the 12-bit quantity m, the contents of the next program address (P + 1), with d to form an 18-bit operand or operand address.



The instruction codes are described in detail in Volume 2 of this reference manual.

ADDRESS MODES

Program indexing can be accomplished and operands can be manipulated in several modes. The two instruction formats provide for 6-bit or 18-bit operands and for 6-bit, 12-bit or 18-bit addresses.

NO ADDRESS

In this mode d or dm is taken directly as an operand. This mode eliminates the need for storing a large number of constants. The d quantity is considered as a 12-bit number, the upper six bits of which are zero. The dm quantity has d as the upper six bits and m as the lower 12 bits.

DIRECT ADDRESS

In this mode, d or m + (d) is used as the address of the operand. The d quantity specifies one of the first 64 addresses in memory ($0000\text{-}0077_8$). The m + (d) quantity generates a 12-bit address for referencing all possible peripheral processor memory locations ($0000\text{-}7777_8$). If $d \neq 0$, the content of address d is added to m to produce an operand address (indexed addressing). If d = 0, m is taken as the operand address. If m = 7777_8, the address is 0. Address 7777_8 is only accessible if the value of d is 7777_8 and m = 7777_8.

EXAMPLE: Address Modes

Given: d = 25
m = 100
contents of location 25 = 0150
contents of location 150 = 7776
contents of location 250 = 1234

Then:

MODE	INSTRUCTION	(A) REGISTER
No Address	14 d	000025
	20 dm	250100
Direct Address	30 (d)	000150
	50 (m + (d))	001234
Indirect Address	40 ((d))	007776

INDIRECT ADDRESS

In this mode, d specifies an address which holds the address of the desired operand. Thus, d specifies the operand address indirectly. Indirect addressing and indexed addressing requires one more memory reference than does direct addressing. Address 7777_8 is accessible if the desired operand address is 7777_8.

ACCESS TO CENTRAL MEMORY

The peripheral processors have access to all central memory storage locations. One word or a block of words can be transferred from a peripheral processor memory to central memory or vice versa. Data from external devices is read into a peripheral processor memory and, with additional instructions, transferred from there to central memory. Conversely, data is transferred from central memory to a peripheral processor memory and then transferred, by additional instructions, to external devices. All addresses sent to central memory from peripheral processors are absolute addresses, rather than relative addresses.

CENTRAL MEMORY READ

The 60 instruction is used to read one word and a 61 instruction is used to read a block of 60-bit central memory words. The central memory words are delivered to a five stage read "pyramid" where they are disassembled into five 12-bit words.

One 12-bit word is transferred to a peripheral processor each microsecond. Because the central memory word is 60 bits long, five microseconds are required for the transfer of each central memory word. It is possible to have four peripheral processors time-sharing the "pyramid" so that the transfer rate can be increased to four central memory words each five microseconds.

If more than four peripheral processors are simultaneously requesting central memory Read operations, the instructions are maintained and are accepted in the order in which they appear when the "pyramid" can accept another peripheral processor, unless one of the peripheral processors has priority (see access priority).

The central memory starting address must be entered in the A register before a Read instruction can be executed. A Load dm (20) instruction may be used for this.

ONE WORD READ

For a one word transfer, the d portion of the Read (60) instruction specifies the following:

d = peripheral processor memory address ($0000\text{-}0077_8$) for the first 12-bit word. The remaining words go to locations d + 1, d + 2, etc.

BLOCK READ

For a block transfer, d and m of the read (61) instruction specify the following:

(d) = the number of central memory words to be transferred. It will be reduced by one for each word transferred.

m = the peripheral processor memory first word address. It will be increased by one for each successive word. (A) is increased by one with the transfer of each word to locate consecutive central memory words.

CENTRAL MEMORY WRITE

The 62 instruction is used for one word and the 63 instruction is used for a block transfer. They assemble 12-bit words into 60-bit words and write them in central memory. Assembly is performed in a write "pyramid" and then transferred to central memory. As is the read "pyramid" it can be time-shared by up to four peripheral processors. Write "pyramid" timing is similar to Read "pyramid" timing.

The starting address in central memory is entered in the A register before the Write instruction is executed.

ONE WORD WRITE

For a one word transfer, the d portion of the Write (62) instruction specifies the following:

d = the peripheral processor memory address ($0000\text{-}0077_8$) of the first 12-bit word. The remaining words are taken from d + 1, d + 2, etc.

BLOCK WRITE

For a block transfer, d and m of the Write (63) instruction specify the following:

(d) = the number of central memory words to be transferred. It is reduced by one for each word transferred.

m = the peripheral processor memory starting address. It is increased by one with the transfer of each word for locating each successive word. (A) is increased by one with the transfer of each word to provide consecutive central memory locations.

ACCESS PRIORITY

Two types of access priority are provided. Placing the Central Memory Access Priority (CMAP) switch in the Program Mode position, one or more peripheral processors may be assigned a priority status by setting bit 2^{17} of its A register. This enables the selected peripheral processors to have preference over other peripheral processors in gaining access to central memory. It also makes it possible for a peripheral processor to interrupt an ECS transfer, which is not otherwise possible. Priority should be assigned to no more than three peripheral processors for operations when ECS is inactive because the value of priorities would thereby be defeated. For operations when ECS is active, priority usage should be limited, because even one interruption of an ECS transfer degrades the transfer rate significantly.

Placing the CMAP switch in the Constant Mode position forces 2^{17} set for all peripheral processors. This makes it possible for any peripheral processor to interrupt an ECS transfer, however, there is no preferential priority among the peripheral processors.

INPUT/OUTPUT

The peripheral equipment connected to the data channels can be accessed by each of the peripheral processors. Input/output instructions select a data channel to contact a unit of peripheral equipment and to initiate transfer of data to or from that equipment. The instructions can determine whether or not a channel (and the peripheral equipment) is available and ready to transfer data.

Each type of peripheral equipment (including a control console) has a set of external function codes which must be used by the peripheral processors for communication with the equipment. These function codes are explained in the applicable reference manual for each type of equipment.

DATA CHANNELS

The number of data channels is dependent on the number of peripheral processors in the system. Each channel has a 12-bit bi-directional data register and two control flags which allow the peripheral processors to monitor the status of the data channels.

CHANNEL ACTIVE/INACTIVE FLAG

When a Function instruction specifies a mode of operation, it places a function word in the channel data register and activates the channel. When the peripheral equipment accepts the function word from the data register, its response clears the data register and the channel active flag.

If an active channel instruction is used with other data transfer instructions, a disconnect channel instruction is required to clear the channel active flag.

REGISTER FULL/EMPTY FLAG

A channel data register is full when it contains a function or data word for an external equipment or contains a word received from an external equipment. The register is empty when it is cleared. The flags are set or clear as the register changes state.

On data output, the peripheral processor places a word in the channel register and sets the full flag. When the external device accepts the word, it clears the register, and clears the full flag.

On data input, the external device places a word in the channel data register and sets the full flag. When the peripheral processor stores the word, it clears the register, and clears the full flag.

DATA INPUT

Several instructions are necessary to transfer data from external equipment into a peripheral processor. The instructions prepare the channel and equipment for the transfer and then start the transfer. Some external equipment, once started, sends a series of words (record) spaced at equal time intervals and then stops between records; Magnetic tape equipment for example. The peripheral processor can read all or a part of the record and then disconnect the channel to end the operation and to make the channel inactive. Other equipment, such as the display console, can send one word (or character) and then stop. The input instructions allow the input transfer to vary from one word to the capacity of the peripheral processor.

An input transfer may be accomplished in the following way:

1. Determine if the channel is inactive. A Jump to m on channel d Inactive (65) instruction does this. Here, m can be a function instruction to select Read mode or determine the status of the equipment.

2. Determine if the equipment is ready. A Function m on Channel d (77) instruction followed by an Active channel d (74) followed by an Input to A from Channel d (70) instruction loads A with the status response of the desire equipment. Here, m is a status request code, and the status response in A can be tested to determine the course of action.

3. Disconnect Channel d (75); this avoids hanging up the procedure.

4. Select Read mode in the equipment. A Function m on Channel d (77) instruction or Function (A) on Channel d (76) instruction will send a code word to the desired device to prepare it for data transfer.

5. Enter the number of words to be transferred in A. A Load d (14) or Load (d) (30) instruction will accomplish this.

6. Activate the channel. An Activate Channel d (74) instruction sets the channel active flag and prepares for the impending data transfer.

7. Start input data transfer. An Input (A) Words to m on Channel d (71) instruction or an Input to A from Channel d (70) instruction starts data transfer. The 71 instruction transfers one word or up to the capacity of the processor memory. The 70 instruction transfers one word only.

8. Disconnect the channel. A Disconnect Channel d (75) instruction makes the channel inactive and stops the flow of input information.

The design of some external equipment requires timing considerations in issuing function, activate, and input instructions. The timing consideration may be based on motion in the equipment, i.e., the equipment must attain a given speed before sending data (e.g., magnetic tape). In general, timing considerations can be ignored by issuing the necessary instructions without an intervening time gap. The external equipment reference manuals list timing considerations which must be taken into account.

DATA OUTPUT

The data output operation is similar to data input in that the channel and equipment must be ready before the data transfer is started by an output instruction.

An output transfer may be accomplished in the following way:

1. Determine if the channel is inactive. A Jump to m on Channel d Inactive (65) instruction does this. Here, m can be a function instruction to select Write mode or determine the status of the equipment.

2. Determine if the equipment is ready. A Function m on Channel d (77) followed by an Activate channel d (74) followed by an Input to A from Channel d (70) instruction loads A with the status response of the desired equipment. Here, m is a status request code, and the status response in A can be tested to determine the course of action.

3. Disconnect Channel d (75); this avoids hanging up the processor.

4. Select Write mode in the equipment. A Function m on Channel d (77) instruction or Function (A) on Channel d (76) instruction will send a code word to the desired device to prepare it for data transfer.

5. Enter the number of words to be transferred in A. A Load d (14) or Load (d) (30) instruction will accomplish this.

6. Activate the channel. An Activate Channel d (74) instruction signals an active channel and prepares for the impending data transfer.

7. Start data transfer. An Output (A) Words from m on Channel d (73) instruction or an Output from A on Channel d (72) instruction starts data transfer. The 73 instruction can transfer one or more words while the 72 instruction transfers only one word.

8. Test for channel empty. A Jump to m if Channel d Full (66) instruction where m = current address, provides this test. The instruction exits to itself until the channel is empty. When the channel is empty, the processor goes on to the next instruction which generally disconnects the channel. The instruction acts to idle the program briefly to ensure successful transfer of the last output word to the recording device.

9. Disconnect the channel. A Disconnect Channel d (75) instruction makes the channel inactive. Data flow in this case terminates automatically when the correct number of words is sent out.

Instruction timing considerations, as in a data input operation, are a function of the external device. Refer to the applicable reference manual for the peripheral equipment timing information.

INTERLOCK REGISTER

The interlock register may be accessed by each of the peripheral processors through a common internal channel.

PROGRAMMING SEQUENCE

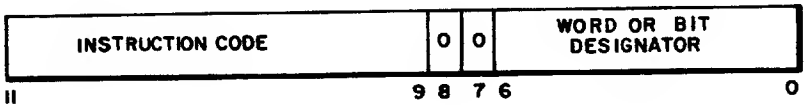
Accessing the interlock register requires a load dm (2000) instruction, a description word (described below), an output on channel 15 (7215) instruction, and an input on channel 15 (7015) instruction.

NOTE

Since channel 15 is always active, bit 5 of the input and output instruction does not function as stated in the instruction description when accessing this channel. In fact, it must not be set as it may allow more than one peripheral processor to access the interlock register at the same time.

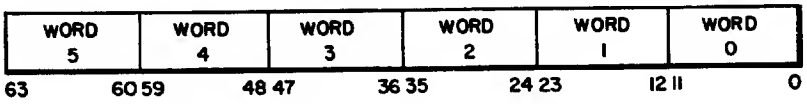
DESCRIPTOR WORD

The descriptor word format is:

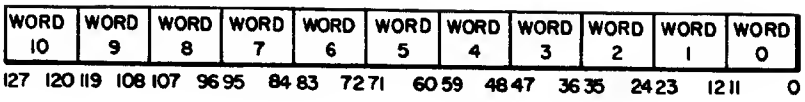


The instruction codes used in the descriptor word are as follows:

0XXX - Read the designated word in the interlock register. There are six words in a 64-bit register and 11 words in a 128-bit register. The words in a 64-bit register are as follows:



The words in a 128-bit register are as follows:



1XXX - Test the designated bit in the interlock register. The status is returned as bit 0 of a 12-bit word. A "1" indicates that the tested bit is set and a "0" indicates that the tested bit is clear.

2XXX - Clear the designated bit in the interlock register. A "0" is reported to the peripheral processor.

3XXX - Test the designated bit and leave it in the clear condition.

4XXX - Set the designated bit. A "0" is reported to the peripheral processor.

5XXX - Test the designated bit and leave it in the set condition.

6XXX - Clear all bits in the interlock register. A "0" is reported to the peripheral processor.

7XXX - Test all bits in the interlock register. The status is returned as a "1" if one or more bits of the interlock register is set.

EXAMPLE: To read word 3, the following sequence would be used:

```
2000    load dm
0003    read word 3
7215    output on channel 15
7015    input on channel 15
```

MANUAL CONTROL

Manual control of system operation is provided through the console or other keyboard. For starting a down system, the Dead Start panel must be used to enter a 12-word program (normally a load routine) to start up operation. The console or other keyboard provides for the entry of data or instructions under program control.

DEAD START PANEL

The three modes of operation, load, sweep, and dump are selectable via the dead start panel; they are described below.

LOAD MODE

To load programs and data into the computer system, the MODE switch must be placed in the LOAD position. The matrix of toggle switches must then be set to a 12-word (or less) program (switch up = "1", switch down = "0"). The program set in the switch matrix should be a load routine to load a larger program from an input device such as a disk file or magnetic tape unit.

Turn the DEAD START switch ON momentarily, then OFF. That initiates the following operations:

1. Assigns all peripheral processors to corresponding data channels.
2. Sends a Master Clear to all I/O channels. A Master Clear removes all equipment selections except the dead start panel, and sets all channels to the Active and Empty condition (ready for input).
3. Sets all peripheral processors to the Input (71) instruction.
4. Clears the P register and sets the A register to 10000_8 in all processors.
5. Transmits a zero word followed by the 12 words from the toggle switches into memory locations 0000 - 0014_8 of peripheral processor 0, and then disconnects data channel 0 causing word 0015_8 of peripheral processor 0 to be zeroed and causing peripheral processor 0 to start execution with the instruction at location 0001.

After the switch matrix program is read from the dead start panel, the panel is automatically disconnected. Processor 0 reads location 0000, adds one to its content, and begins executing the program at address 0001. The other processors are still set to the Input (71) instruction and may receive data from processor 0 via their assigned channels.

SWEEP MODE

Placing the MODE switch in the SWEEP position and momentarily turning on the DEAD START switch results in the following:

1. Sets all processors to instruction 50X.
2. Clears all processor P registers to zero.

The translation of the 50X instruction in each processor causes each processor to sweep through its memory, reading and restoring the contents of each location, without executing instructions. Sweep mode is a maintenance tool useful in checking the operation of memory logic.

DUMP MODE

Placing the MODE switch in the DUMP position and momentarily turning on the DEAD START switch initiates the following operations:

1. Assigns all peripheral processors to corresponding data channels.
2. Sends a Master Clear to all I/O channels except channel 0.
3. Holds channel 0 to Active and Empty.
4. Sets all processors to the Output (73) instruction.
5. Clears the P register and sets the A register to 10000_8 in all processors.

Each of the processors senses the Active and Empty condition of its assigned channel and outputs the content of its memory address zero. Each of the I/O channels is then set to Full (except channel 0), and the processors wait for an Empty signal. Each processor advances its P register by one and reduces the content of its A register by one (to 7776_8). At this point, the processors waiting for an Empty signal are hung up and cannot proceed.

Channel 0 (assigned to processor 0) is held to Empty by the DUMP position. Processor 0, therefore, proceeds through the 73 instruction until the contents of A are reduced to one. Processor 0 has now dumped its entire memory content on channel 0 (although no I/O device was selected to receive it). Execution then starts with the instruction at the location specified by the contents of location 0000 plus one; it is now free to execute a dump program which must have been previously stored in its memory (location 0000 must have been previously set to the starting address minus one).

PROGRAM/CONSTANT MODE

Placing the CMAP switch in the Program Mode position provides program selectable priority for each peripheral processor. Placing it in the Constant Mode position assigns priority status to all peripheral processors by which any one can interrupt an ECS transfer.

CONSOLE

The display console consists of two cathode ray tube (CRT) display screens and a keyboard for manual entry data.

CONSOLE CONTROLS

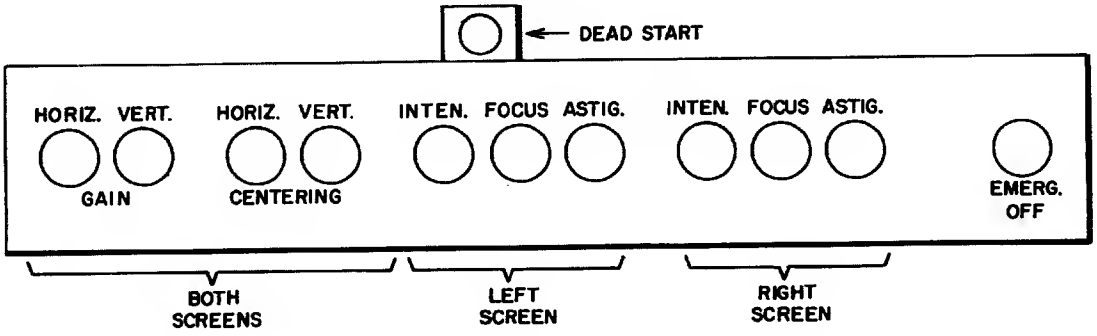


Figure 2-5. Console Operator Control Panel

POWER ON/OFF Switch (located under the right side of the desktop.)
This switch applies or disconnects the console ac power.

HORIZONTAL GAIN Control
This control varies the width of the displays.

VERTICAL GAIN Control
This control varies the height of the displays.

HORIZONTAL CENTERING Control
This control varies the horizontal location of the displays.

VERTICAL CENTERING Control
This control varies the vertical location of the displays.

INTENSITY
These two controls vary the brightness of the displays.

FOCUS
These two controls are used to obtain image clarity in the center area of the displays.

ASTIGMATISM Controls
These two controls are used to obtain image clarity at the edges of the displays.

DEAD START Switch
This pushbutton switch dead starts the mainframe.

EMERGENCY OFF Switch
This pushbutton switch immediately disconnects ac power from the display console and the entire mainframe.

CAUTION

This switch removes all system power and does not allow proper mainframe refrigeration system pump down. Unless the system is to be restarted within a few minutes, call customer engineering so that they can perform the pump down.

OPERATING PROCEDURES

To turn the console on, rotate both INTENSITY controls fully counterclockwise and press the POWER ON/OFF switch to the ON position.

CAUTION

Failure to rotate INTENSITY controls fully counterclockwise prior to warm-up may result in irreparable damage to the CRT's.

After the built-in 40- to 80-second time delay, rotate the INTENSITY controls clockwise to obtain proper character intensity. In the event it is necessary to turn the console off, rotate both INTENSITY controls fully counterclockwise and press the POWER ON/OFF switch.

NOTE

See Volume 2 of this reference manual for program interaction and keyboard usage.

SYSTEM INTERRUPT

Detecting and handling interruptible conditions involves both hardware and software. This section describes hardware provisions for detecting and handling interrupt. The features of an operating system used for implementing interrupts are described in the operating system reference manuals.

HARDWARE PROVISIONS FOR INTERRUPT

EXCHANGE JUMP

Within a peripheral processor, execution of an Exchange Jump instruction initiates hardware action in the central processor to interrupt the current central processor program and substitute another program, the parameters of which are defined in the Exchange Jump package. The Exchange Jump is also used to start the central processor from a Stop condition.

CHANNEL AND EQUIPMENT STATUS

Within the peripheral processors, hardware flags indicate the state of various conditions in the data channels, e.g., Full/Empty, and Active/Inactive. External equipment devices are capable of detecting certain errors (e.g., parity error) and holding status information reflecting their operating conditions (e.g., Read, End of File, etc.) Channel and equipment status information may be examined by instructions in the peripheral processors. The Input/Output section describes these instructions. For detailed status information on external devices such as magnetic tape units and card readers, refer to the applicable reference manual for each device or its controller.

EXIT MODE

Central processor hardware provides for three types of error halt conditions (Exit mode):

- Address out of range (i.e., out of bounds)
- Operand out of range (i.e., exponent overflow)
- Indefinite result

Detecting the occurrence of one or more of these conditions is accomplished by the hardware and causes an error exit. Note that halting on any of these conditions is selectable.

TIMING INFORMATION

Instruction execution times are explained in this section. The basic times are listed in tables, however there are certain conditions which must be taken into account to permit calculation of program execution timing as follows:

CENTRAL PROCESSOR TIMING

The instructions and their execution times are shown in Table 2-5.

1. A minimum of 5 minor cycles is required from the end of Increment unit time until the next operand is retrieved from central memory and is ready in the X register. Memory bank conflicts can lengthen the time.
2. Memory bank conflicts cause a delay of at least 3 minor cycles. The address is issued every 3 cycles until it is accepted by memory.

3. Instructions are issued from the instruction stack at a maximum rate of one 60-bit instruction word every 4 minor cycles. If the instruction word comes from memory the same rate applies except for memory conflicts. If a conflict occurs the maximum rate becomes one word every 8 minor cycles.

4. The time required from the issue of the last parcel of an instruction word until the issue of the first parcel of the next instruction word is one minor cycle whether the word comes from the stack or from memory. If from memory, memory conflicts can cause the time elapsed to be 4 minor cycles. If the second parcel of an instruction word is a 30-bit instruction, add one more cycle to the times specified.

5. Operand registers are available to more than one function unit in a single minor cycle if the function units are from different groups. The groups are as follows:

GROUP 1	GROUP 2	GROUP 3
Boolean	Shift	
Divide	Floating Add	Increment 1
Multiply 1	Long Add	Increment 2
Multiply 2		

6. A result register can be used as an operand register in the same minor cycle.

TABLE 2-5. CENTRAL PROCESSOR INSTRUCTION EXECUTION TIMES

		CPU-0	CPU-1	NOTES
00XXX	Error exit to MA or Program Stop			4
0100K	Return jump to K	13	21	6
011jK	Read extended core storage	-	-	2
012jK	Write extended core storage	-	-	2
013jK	Central exchange jump	-	-	-
02ixK	Jump to (Bi) + K	14	15	9, 10
030jK	Jump to K if (Xj) = 0	9	15	1, 3, 6, 9
031jK	Jump to K if (Xj) ≠ 0	9	15	1, 3, 6, 9
032jK	Jump to K if (Xj) positive	9	15	1, 3, 6, 9
033jK	Jump to K if (Xj) negative	9	15	1, 3, 6, 9
034jK	Jump to K if (Xj) in range	9	15	1, 3, 6, 9
035jK	Jump to K if (Xj) out of range	9	15	1, 3, 6, 9
036jK	Jump to K if (Xj) definite	9	15	1, 3, 6, 9
037jK	Jump to K if (Xj) indefinite	9	15	1, 3, 6, 9
04ijK	Jump to K if (Bi) = (Bj)	8	15	1, 3, 6, 9, 10
05ijK	Jump to K if (Bi) ≠ (Bj)	8	15	1, 3, 6, 9, 10
06ijK	Jump to K if (Bi) ≥ (Bj)	8	15	1, 3, 6, 9, 10
07ijK	Jump to K if (Bi) < (Bj)	8	15	1, 3, 6, 9, 10

TABLE 2-5. CENTRAL PROCESSOR INSTRUCTION EXECUTION TIMES (Cont'd)

		CPU-0	CPU-1	NOTES
10ijk	Transmit (Xj) to Xi	3	4	
11ijk	Logical product of (Xj) and (Xk) to Xi	3	5	
12ijk	Logical sum of (Xj) and (Xk) to Xi	3	4	
13ijk	Logical difference of (Xj) and (Xk) to Xi	3	5	
14i0k	Transmit complement of (Xk) to Xi	3	4	
15ijk	Logical product of (Xj) and comp (Xk) to Xi	3	5	
16ijk	Logical sum (Xj) and comp (Xk) to Xi	3	4	
17ijk	Logical difference of (Xj) and comp (Xk) to Xi	3	5	
20ijk	Left shift (Xi) by jk	3	6	
21ijk	Right shift (Xj) by jk	3	6	
22ijk	Left shift (Xk) nominally (Bj) places to Xi	3	6	
23ijk	Right shift (Xk) nominally (Bj) places to Xi	3	6	
24ijk	Normalize (Xk) to Xi and Bj	4	7	
25ijk	Round and normalize (Xk) to Xi and Bj	4	7	
26ijk	Unpack (Xk) to Xi and Bj	3	7	
27ijk	Pack Xi from (Xk) and (Bj)	3	7	
30ijk	Floating sum of (Xj) and (Xk) to Xi	4	11	
31ijk	Floating difference of (Xj) and (Xk) to Xi	4	11	
32ijk	Floating DP sum of (Xj) and (Xk) to Xi	4	11	
33ijk	Floating DP difference of (Xj) and (Xk) to Xi	4	11	
34ijk	Round floating sum of (Xj) and (Xk) to Xi	4	11	
35ijk	Round floating difference of (Xj) and (Xk) to Xi	4	11	
36ijk	Integer sum of (Xj) and (Xk) to Xi	3	6	
37ijk	Integer difference of (Xj) and (Xk) to Xi	3	6	
40ijk	Floating product of (Xj) and (Xk) to Xi	10	57	
41ijk	Round floating product of (Xj) and (Xk) to Xi	10	57	
42ijk	Floating DP product of (Xj) and (Xk) to Xi	10	57	
43ijk	Form mask in Xi, jk bits	3	6	
44ijk	Floating divide (Xj) by (Xk) to Xi	29	57	
45ijk	Round floating divide (Xj) by (Xk) to Xi	29	57	
46(oxx)xx	No operation (pass)	1	3	
47ixk	Count the numbers or "1's" in (Xk) to Xi	8	68	
50ijK	Set Ai to (Aj) + K	3	-	8, 7
51ijK	Set Ai to (Bj) + K	3	-	8, 7
52ijK	Set Ai to (Xj) + K	3	-	8, 7
53ijk	Set Ai to (Xj) + (Bk)	3	-	8, 7
54ijk	Set Ai to (Aj) + (Bk)	3	-	8, 7
55ijk	Set Ai to (Aj) - (Bk)	3	-	8, 7
56ijk	Set Ai to (Bj) + (Bk)	3	-	8, 7
57ijk	Set Ai to (Bj) - (Bk)	3	-	8, 7
60ijK	Set Bi to (Aj) + K	3	5	
61ijK	Set Bi to (Bj) + K	3	5	
62ijK	Set Bi to (Xj) + K	3	5	
63ijk	Set Bi to (Xj) + (Bk)	3	5	
64ijk	Set Bi to (Aj) + (Bk)	3	5	
65ijk	Set Bi to (Aj) - (Bk)	3	5	
66ijk	Set Bi to (Bj) + (Bk)	3	5	
67ijk	Set Bi to (Bj) - (Bk)	3	5	
70ijK	Set Xi to (Aj) + K	3	6	
71ijK	Set Xi to (Bj) + K	3	6	

TABLE 2-5. CENTRAL PROCESSOR INSTRUCTION EXECUTION TIMES (Cont'd)

		CPU-0	CPU-1	NOTES
72ijK	Set Xi to (Xj) + K	3	6	
73ijk	Set Xi to (Xj) + (Bk)	3	6	
74ijk	Set Xi to (Aj) + (Bk)	3	6	
75ijk	Set Xi to (Aj) - (Bk)	3	6	
76ijk	Set Xi to (Bj) + (Bk)	3	6	
77ijk	Set Xi to (Bj) - (Bk)	3	6	

NOTES:

1. Add 6 cycles for a jump out of the stack for an instruction. Add 2 cycles for a no jump condition. Add 5 cycles for a no jump condition out of the stack for CPU-0.
2. Refer to the ECS Description manual (See Preface) for timing information.
3. For CPU-1, jumps in which the jump conditions are not present, add 2 cycles.
4. If error exit to MA is selected, cycles.
5. All times are in minor cycles (100 nsec).
6. Times do not account for memory conflicts which can cause delays. See text.
7. For CPU-0, the A register is reserved for 3 cycles. When i = 1-5, Xi is reserved for 8 cycles. When i = 6 or 7, Xi is reserved for 9 cycles.
8. For CPU-1, when i = 0, 6 cycles, when i = 1-5, 14 cycles, when i = 7 or 8, 12 cycles.
9. When more than one functional unit is used, both must be free before the instruction can be issued.
10. Increment unit time affects timing, see text.

PERIPHERAL PROCESSOR TIMING

The instructions are listed in Table 2-6. Certain considerations which might affect the times shown in the table are:

1. Instructions with the 24-bit format require 10 extra cycles (1 major cycle) to read m. These are the indirect and indexed addressing instructions.
2. Instructions for input/output and for memory references can transfer a word every 10 cycles although the peripheral equipment seldom permits this rate for input/output operations.
3. Conflicts with the central processor for central memory references cause indeterminate delays.
4. Following an Exchange Jump instruction, the central processor must complete the exchange jump before further memory references or Exchange Jump instructions can be executed.
5. In systems with 14, 17, or 20 peripheral processors, certain delays occur because the data channels are mounted in an external cabinet. A delay of four minor cycles occurs in recognizing a changed channel status. A peripheral processor can not

recognize a change in status of a data channel made by any of the four processors preceding it in the "barrel". For example; after a channel goes inactive, the next four processors in succession do not recognize the change, so the fifth peripheral processor is the first one that can recognize and take advantage of the change.

TABLE 2-6. PERIPHERAL PROCESSOR INSTRUCTION EXECUTION TIMES

		CYCLES	NOTES
00	Pass	10	
01	Long jump to m + (d)	-	1
02	Return jump to m + (d)	-	2
03	Unconditional jump d	10	
04	Zero jump d	10	
05	Nonzero jump d	10	
06	Plus jump d	10	
07	Minus jump d	10	
10	Shift d	10	
11	Logical difference d	10	
12	Logical product d	10	
13	Selective clear d	10	
14	Load d	10	
15	Load complement d	10	
16	Add d	10	
17	Subtract d	10	
20	Load dm	20	
21	Add dm	20	
22	Logical product dm	20	
23	Logical difference dm	20	
24	Pass	10	
25	Pass	10	
260X	Exchange jump	-	3
261X	Monitor exchange jump	-	3
262X	Monitor exchange jump to MA	-	3
27	Read program address	10	
30	Load (d)	20	
31	Add (d)	20	
32	Subtract (d)	20	
33	Logical difference (d)	20	
34	Store d	20	
35	Replace add (d)	30	
36	Replace add one (d)	30	
37	Replace subtract one (d)	30	
40	Load (d)	30	
41	Add ((d))	30	
42	Subtract ((d))	30	
43	Logical difference ((d))	30	
44	Store ((d))	30	
45	Replace add ((d))	40	
46	Replace add one ((d))	40	
47	Replace subtract one ((d))	40	

TABLE 2-6. PERIPHERAL PROCESSOR INSTRUCTION EXECUTION TIMES (Cont'd)

		CYCLES	NOTES
50	Load (m + (d))	-	2
51	Add (m + (d))	-	2
52	Subtract (m + (d))	-	2
53	Logical difference (m + (d))	-	2
54	Store (m + (d))	-	2
55	Replace add (m + (d))	-	4
56	Replace add one (m + (d))	-	4
57	Replace subtract one (m + (d))	-	4
60	Central read from (A) to d	-	5
61	Central read (d) words to (A) from m	-	6
62	Central write to (A) from d	-	5
63	Central write (d) words to (A) from m	-	6
64	Jump to m if channel d active	20	
65	Jump to m if channel d inactive	20	
66	Jump to m if channel d full	20	
67	Jump to m if channel d empty	20	
70	Input to A from channel d	20	
71	Input (A) words to m from channel d	-	7
72	Output from A on channel d	20	
73	Output (A) words from m on channel d	-	7
74	Activate channel d	20	
75	Disconnect channel d	20	
76	Function (A) on channel d	20	
77	Function m on channel d	20	

NOTES:
1. 30 cycles unless d = 0, then 20 cycles
2. 40 cycles unless d = 0, then 30 cycles
3. 26 cycles because of central memory access limitations (If exchange is executed) If exchange is not executed it will be 10 cycles.
4. 50 cycles unless d = 0, then 40 cycles
5. Minimum of 60 cycles
6. 50 cycles plus 50 cycles per word
7. 40 cycles plus 10 cycles per word
8. All times are in minor cycles (100 nsec)

COMMENT SHEET

MANUAL TITLE CDC CYBER 70 Model 74 Volume 1 System Description and Programming Information Reference Manual

PUBLICATION NO. 60347400 REVISION N

FROM: NAME: _____

BUSINESS ADDRESS: _____

CUT ALONG LINE

PRINTED IN U.S.A.

AA3419 REV. 7/75

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY
CONTROL DATA CORPORATION
Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD

FOLD